ARMY RESEARCH LABORATORY

# OneSAF Test Bed (OTBSAF) Automation

**by Ronald D. Anderson and My Van Hoang Baranoski**

**ARL-TN-0242** **May 2005**

## NOTICES

### Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

DESTRUCTION NOTICE—Destroy this report when it is no longer needed. Do not return it to the originator.

# Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

**ARL-TN-0242** **May 2005**


# OneSAF Test Bed (OTBSAF) Automation


**Ronald D. Anderson and My Van Hoang Baranoski**
**Weapons and Materials Research Directorate, ARL**

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED (From - To) |
|---|---|---|
| May 2005 | Final | November to December 2004 |

**4. TITLE AND SUBTITLE**
OneSAF Test Bed (OTBSAF) Automation

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**
Ronald D. Anderson and My Van Hoang Baranoski (both of ARL)

**5d. PROJECT NUMBER**
622618H8011

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
Weapons and Materials Research Directorate
Aberdeen Proving Ground, MD 21005-5066

**8. PERFORMING ORGANIZATION REPORT NUMBER**
ARL-TN-0242

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

A method of automating the OTBSAF (OneSAF Test Bed) combat simulation program is described, including source code additions, installation steps, and a sample run. The programmed stop criteria are based on simulation time and combat vehicle damage. These new functions allow the program operator to prepare and complete many simulations without constant monitoring and intervention, thus saving man-hours while a suite of runs necessary to gain required confidence levels of results is completed. Output information includes a time-sequenced list of vehicle status conditions and the final stop criterion.

**15. SUBJECT TERMS**
OneSAF; OTBSAF; semi-automated forces; simulation

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Ronald D. Anderson |
| Unclassified | Unclassified | Unclassified | SAR | 50 | 19b. TELEPHONE NUMBER (*Include area code*) 410-278-6102 |

**Standard Form 298 (Rev. 8/98)**
**Prescribed by ANSI Std. Z39.18**

## Contents

# 1. Introduction

Combat simulation programs often use probabilities to determine results of actions, such as how much damage will be inflicted upon a vehicle when it is hit by an enemy round or whether an observer will notice a distant enemy vehicle within a given time period. If the probabilities are assigned through a random process (or Monte Carlo function), then the outcomes are also random. The results of such a study are given as probabilities of outcomes within certain confidence levels. The certainty of a confidence level is determined through many iterations of the simulation. When a simulation program has no capability for automatic initiation or termination, the scientist responsible for the statistical analysis is often required to manually start and monitor the program for each iteration—even when there is no need for human-computer interaction during the program execution other than initial setup.

The OTBSAF (*1*) program (OneSAF [semi-automated forces] Test Bed) uses Monte Carlo methods to determine probabilities of the actions and results of combat. Therefore, a single combat result is seldom calculated for more than a given percentage of OTBSAF simulations, even when initial conditions are the same. Many program executions are commonly required to determine statistical confidence in a result. OTBSAF, however, has no automatic provision for ending combat simulations; once started, OTBSAF requires the operator to stop the simulation when s/he judges that some set of termination parameters has been met.

During the simulation, the operator can "save" current conditions into a computer file which may be used to set up a new simulation at a later time, even the next day or month. When the conditions are re-loaded, the saved file does the initialization and the operator may then simply monitor the execution (through the graphical user interface [GUI]) to determine at a later time when the simulation has progressed enough to determine combat results. Then the same initialization may be done again and a new calculation may be started, perhaps to end in a different result some time later. After dozens of restarts, the operator may have enough statistical results to determine probability of a certain outcome and its confidence level.

Features to allow "unmanned" iterations of OTBSAF were created so that multi-run studies could be easily accomplished without work stations being constantly attended during the simulations. The two major features supplied are a maximum run-time value and a method of stopping execution based on kill levels of vehicles. Both features are user implemented via the OTBSAF execution statement.

## 2. Changes in Source Code – Maximum Run Time Shutoff

### 2.1 Subdirectory src/OTBSAF

The *main.c* code and *main.h* header (appendices A and B, respectively) are modified to allow an extra execution line input value at run time. The *main.h* header file defines the options structure, and parameters are added with type definitions and default values.

The maximum run time parameter is named "run_duration" and may be added to the execution line as the option "-run_duration xxx" where the xxx is an integer value of milliseconds to be compared to internal program time of execution as the run progresses. A call within *main.c* to the scheduler subroutine initiates execution of the "*main_clean_up*" routine at time "xxx". When run time is equal to or greater than the entered value, *main_clean_up* shuts down the program gracefully, closing necessary files and removing any temporary data stored from the program execution. Default value of the shut-down time is 0 (zero), which is used to signify that the program does not schedule a function call to the *main_clean_up* routine.

The number of milliseconds to enter for stopping a simulation depends upon how long the operator determines the combatants will require to accomplish the assigned missions. Usually, the best way to establish this value is to observe an example of the study and note the program start and end times. Adding 25% to 50% more time for the scenario should give enough time for most variations to finish.

This function was developed by the OneSAF team at U.S. Armament Research, Development, and Engineering Center, Picatinny, Arsenal, New Jersey (*2*).

## 3. Changes in Source Code – Shut-off Based on Vehicle Kill Percentage

### 3.1 Subdirectory src/OTBSAF

The parameters that control program shut-down because of damage of vehicles are "bpct," "rpct," "bk_lev," and "rk_lev". Again, these are execution line parameters, which are used as in "-bpct 50," where the parameter name is always preceded by a "-" sign and the value is interpreted as an integer. To enter percentage levels of vehicles (blue and red armies, respectively) killed during the run execution, "bpct" and "rpct" are used. The value "50" stands for "50%" of combatants killed. When the percentage level of killed vehicles within the designated army reaches or exceeds the desired percentage, the program will generate a call to *main_clean_up* and terminate.

Kill levels “bk_lev” and “rk_lev” are used to compare with vehicles as they are checked for damage. Levels are defined as 1 through 4, where level 4 requires the vehicles to sustain catastrophic damage in order to be counted killed. Level 3 allows either catastrophic or mobility and firepower (M&F) damage for counting kills. Level 2 allows any level of damage that includes firepower impairment to be counted as a kill. Level 1 includes any vehicle whose damage level is other than “healthy” to be counted as a kill. If “rk_lev” is 3 and “bk_lev” is zero, then the blue army’s vehicle damage will not be part of the run stop tests, but red army vehicles suffering catastrophic or M&F kills will be counted. If “rk_lev” is zero and “bk_lev” is also zero, then no run termination will be done for vehicle damage counts, no matter what values have been entered for “bpct” and “rpct”. Default values are all 0 (zero).

The changes in OTBSAF source code in the *src/OTBSAF* directory are in *main.c* and *main.h* (see appendices C and D). As with “run_duration,” the four parameters are defined in the data structure “main_options_struct” within *main.h*. Code in *main.c* copies values from the structure into like-named parameters within a file *libkillstop.h* (appendix E) included in the headers for the *main.c* routine.

### 3.2 Subdirectory libsrc/libentity

Two files were written for this directory to look for vehicle kill levels, and both *check_list.c* and *print_vehicle_info.c* (appendices F and G) are defined as externals in the *libentity.h* (appendix H) header file. The subroutine *ent_tick* (in file *ent_tick.c*; appendix I) calls *check_list* during program execution with vehicle identification (ID) values for all known vehicles during the OTBSAF run. A dynamically linked list of vehicle IDs is built by *check_list* for comparison as the OTBSAF run progresses. When a new valid vehicle ID is found, *check_list* performs two tasks: it calls *sched_periodic_fncl* to schedule a once-per-minute entry to *print_vehicle_info* for determining vehicle suitability and damage level, and it adds a new node containing the new vehicle ID to the linked list. Besides individual vehicles, aggregate vehicle formations (platoons, companies, etc.) also have vehicle IDs sent to *print_vehicle_info.* To avoid counting these aggregate entities, the linked list entry is tested for this condition and the function exits before the status of non-single vehicle IDs is examined.

Subroutine *print_vehicle_info* is called once per minute per vehicle, so the vehicle status is evaluated only once each 60 seconds of OTBSAF run time. The *print_vehicle_info* routine keeps track of the integer minute value of the system clock during each call; as long as the minute value remains the same for successive entries, the total number of blue (or red) vehicles is increased with each call. At each call, if the vehicle is damaged, at least “bk_lev” (or “rk_lev”) an additional killed vehicle counter is increased. At the first instance when the minute value does not match the previous value, we determine a percentage of army kill by multiplying the counted kills by 100 and dividing by the total number of blue or red army vehicles. The result is compared to “bpct” or “rpct” to test whether to stop the OTBSAF run. If the killed percentage is too low, the vehicle and kill counts are set to 0 (zero) and counting begins again for

the current minute. If the damaged vehicle percentage matches or exceeds the "bpct" or "rpct" value, then *print_vehicle_info* initiates a call to the *main_clean_up* routine to stop the simulation.

Subroutine *print_vehicle_info* also writes current vehicle and kill counts into a local file called "VehicleInfo" (example in appendix J) during each damage test. If this file is not removed after the OTBSAF run stops, new information from subsequent run(s) will be appended.

A header file called *libkillstop.h* in the *libsrc/libentity* directory is used to define the four kill-related values to *print_vehicle_info.c* (and to *main.c* from the *src/OTBSAF* directory). The header file needs to be copied to the *include/libinc* directory so it can be accessed by both the *main* and *print_vehicle_info* object files during program linking; we do this by defining *libkillstop.h* as a header in the *GNUmakefile.in* source code (printed in appendix K) and in its derivative *GNUmakefile*. Also, *check_list.o* and *print_vehicle_info.o* are added to the list of object files in the *GNUmakefile.in* coding.

### 3.3 Additional Files

In order to manage automated simulations, two files are needed for program initiation and execution. The first file is a controller containing execution statements and pertinent execution line parameters. A sample execution line may look like this:

./otbsaf –nonet –run_duration 300000 –rpct 75 –rk_lev 3 –parser –sourcefile ./inpt

In this situation, the maximum run time is set at 5 minutes (300,000 milliseconds), and a stop feature is set for red vehicles reaching a 75% kill percentage (kill at level 3 is defined as either catastrophic damage or a combination of M&F damage). The program takes its initial setup via information parsed from a second file (here called "./inpt") containing OTBSAF commands to reference a pre-stored scenario of starting positions for vehicles and other OTBSAF objects. The input might be:

scenario load RDA.1.gz

in which scenario RDA.1 is the name of the scenario and the information is stored in an archived file which the program calls by its full name "RDA.1.gz" in order to initialize the simulation. The scenario should also contain vehicle mission data, since the operator will not be able to enter commands during an unattended run. Other commands may also be given in this file, such as

run 2.0 0

which directs the program to run at twice the normal execution speed. An example of a parser information file is printed in appendix L.

To run OTBSAF in background, one should use the Unix[1] "at" command to schedule a run (or a series of runs) at a later time. Using the "&" to put an OTBSAF run in current background execution does not work well, as the program may hang up during initialization.

# 4. Adding Features to OTBSAF Version 1

Although only two subdirectories receive new source code, several other OTBSAF locations are changed at compilation time. The steps necessary to properly add the features include the following.

## 4.1 Subdirectory *libsrc/libentity*

Insert the new source code containing definitions of the external object codes *check_list.o* and *print_vehicle_info.o* into *libentity.h* so that compiled code can reference them.

In *ent_tick.c*, add the new code calling *check_list* with vehicle ID numbers during simulation execution.

Add *check_list.c*, *print_vehicle_info.c*, and *libkillstop.h* as completely new files.

In *GNUmakefile* and *GNUmakefile.in*, add the code referencing the new object files and header file.

Execute a "gmake clean," then "gmake all" in the *libentity* directory to recompile all source code and to copy the new header files into the *include/libinc* directory. If compiling finishes without error, the new object codes will be archived as *libentity.a* into the *lib* subdirectory.

## 4.2 Subdirectory *src/OTBSAF*

Insert new source code into *main.c* and *main.h* containing definitions of the new parameters and entering values for them into the *libkillstop.h* header variables.

Execute "gmake otbsaf" to create the *main.o* object file and link it to all the other object files from the *lib* subdirectory.

Use *otbsaf* for an interactive simulation, or add the execution line parameters to create a halt to execution when run time reaches a maximum value or when vehicle kill percentages reach a prescribed limit.

Create a batch run file to automatically start and stop non-interactive simulations, such as the script *doit* in appendix M. In this example, a file named *inpt* contains information defining the

[1]Unix is a trademark of Bell Laboratories.

scenario to be loaded. Examples of output files from an OTBSAF simulation are in file *out1* and *VehicleInfo1* (appendices N and J, respectively); the looping feature in the controller file causes output and *VehicleInfo* file names to have the final digit increased with each simulation. Note *out1* lists all the execution line input, including values for kill percentages and levels and for a maximum run time limit.

## 5. Summary

The additional features allow multiple OTBSAF simulations without user intervention. Output files may be scanned to determine combat results and statistical information from the automatic simulations.

Using the vehicle kill percentage method to stop simulations should also require a run time termination value, since it cannot be guaranteed that the desired percentage of vehicles will be killed (or damaged) before combatants pass each other in terrain or before all available ammunition is expended.

## 6. References


1. OTBSAF, Version 1, Lockheed Martin Information Systems, Martin Marietta Technologies, Inc., 12506 Lake Underhill Road, Orlando FL, under STRICOM Advanced Distributed Simulation II Contract Number N61339-96-D-0002, Delivery Order 97, September 1998.
2. Matyola, Maryann. *OTBReflector Communication*; AMSTA-AR-FS-H; U.S. ARDEC: Picatinny Arsenal, NJ, 3 January 2003.

INTENTIONALLY LEFT BLANK

## Appendix A. *main.c* Code Changes for Maximum Run Time Feature

```
.
.
.
PARSE_TABLE *main_table_ptr;

struct main_options_struct main_options = {
      /* start of new code */
    {
    "Run Duration", "Specifies the length of OTB run in milliseconds",
    NULL, CMD_INTEGER, "run_duration", NULL, 0, 0
    },
      /* end of new code */
.
.
.
int main(
    int    argc,
    argv_t argv)
{
    int status = main_init(argc, argv);

      /* start of new code */
    if (main_options.run_duration.value)
      sched_deferred_fncl((SCHED_FUNCTION)main_clean_up,
      main_options.run_duration.value, 0, A_END);
      /* end of new code */
.
.
.
```

INTENTIONALLY LEFT BLANK

## Appendix B. *main.h* Changes for Maximum Run Time Feature

```
.
.
.
extern struct main_options_struct
{
CMD_INTEGER_OPTION run_duration;                /* new code */
#ifdef USE_MOTIF
    CMD_TOGGLE_OPTION  gui;
    CMD_TOGGLE_OPTION  guiwarnings;
.
.
.
```

INTENTIONALLY LEFT BLANK

## Appendix C. *main.c* Code Change for Vehicle Kill Percentage Termination

```
.
.
.
#include <libvterrain.h>
#include <libkillstop.h>                          /* new code */
.
.
.
struct main_options_struct main_options = {
      /* start of new code */
    {
    "Blue Kill Percentage","Specifies percent of blue vehicles killed for run
stoppage",
    NULL, CMD_INTEGER, "bpct", NULL, 0, 0
    },
    {
    "Red Kill Percentage","Specifies percent of red vehicles killed for run
stoppage",
    NULL, CMD_INTEGER, "rpct", NULL, 0, 0
    },
    {
    "Blue Kill Level","Specifies level of kill for blue vehicles for run
stoppage",
    NULL, CMD_INTEGER, "bk_lev", NULL, 0, 0
    },
    {
    "Red Kill Level","Specifies level of kill for red vehicles for run
stoppage",
    NULL, CMD_INTEGER, "rk_lev", NULL, 0, 0
    },
      /* end of new code */
.
.
.
int main(
    int    argc,
    argv_t argv)
{
    int status = main_init(argc, argv);
.
.
.
    if (status)
      return status;

    rpct = main_options.rpct.value;                  /* new code */
    bpct = main_options.bpct.value;                  /* new code */
    bk_lev = main_options.bk_lev.value;              /* new code */
    rk_lev = main_options.rk_lev.value;              /* new code */
.
.
.
```

INTENTIONALLY LEFT BLANK

## Appendix D. *main.h* Code Changes for Vehicle Kill Percentage Termination

```
.
.
.
extern struct main_options_struct
{
CMD_INTEGER_OPTION bpct;                        /* new code */
CMD_INTEGER_OPTION rpct;                        /* new code */
CMD_INTEGER_OPTION bk_lev;                      /* new code */
CMD_INTEGER_OPTION rk_lev;                      /* new code */
.
.
.
```

INTENTIONALLY LEFT BLANK

## Appendix E. *libkillstop.h* Header File

```
/* libkillstop.h */

/* header to hold values for determining whether to stop computation */
/*    because of percentage killed vehicles (red or blue) */

        int32    bpct,
                 rpct,
                 bk_lev,
                 rk_lev;

/*  bpct   =  cutoff percentage when blue kills reach this level  */
/*  rpct   =  cutoff percentage when red kills reach this level   */
/*  bk_lev =  definition of blue kill                             */
/*         1 = kill when at least mobility is disabled            */
/*         2 = kill when at least firepower is disabled           */
/*         3 = kill when at least mobility and firepower are disabled*/
/*         4 = kill when catastrophic damage                      */
/*  rk_lev =  definition of red kill                              */
```

INTENTIONALLY LEFT BLANK

## Appendix F. *check_list.c* Function Source Code

```
#include "libent_local.h"
#include <stdlib.h>
#include <libsched.h>
/* #include <liblocale.h> */
#include <stdext.h>
#include <libclass.h>
#include <libtime.h>

/* Structure for the nodes of the dynamically linked list */
typedef struct list_node {
        ForceID     my_force;
        VehicleMarking  my_marking;
        struct list_node *link;
}List_Node;

typedef List_Node *list_pointer;

void check_list (int32 vehicle_id)
{
    int32 found_marking = 0;
    VehicleMarking  marking;
    ForceID       force;
    static list_pointer my_list = NULL;
    list_pointer newNodePtr = NULL;
    list_pointer currPtr = NULL;
    list_pointer checklist = my_list;
    ent_get_marking(vehicle_id,&marking);
    force = ent_get_force_id(vehicle_id);

    if (checklist) // if the list is not empty
    {
      found_marking = 0;
        while (checklist != NULL)
        {
            if (strcmp(marking.text,checklist->my_marking.text) == 0)
                {
                  if (force == checklist->my_force)
                  {
                        found_marking = 1;
                              break;
                  }
            }
                currPtr = checklist;
                checklist = checklist->link;
      }
      if (found_marking != 1)
        {

      sched_periodic_fncl(print_vehicle_info,time_last_simulation_clock+5000,
60000,747,A_INT,vehicle_id,A_END);
                newNodePtr = (list_pointer)malloc(sizeof(List_Node));
```

```
                newNodePtr->my_force = force;
                newNodePtr->my_marking = marking;
                newNodePtr->link = NULL;
                currPtr->link = newNodePtr;
            }
        }
        else // if the list IS empty
        {

sched_periodic_fncl(print_vehicle_info,time_last_simulation_clock+5000,60000,
747,A_INT,vehicle_id,A_END);
            my_list = (list_pointer)malloc(sizeof(List_Node));
            my_list->my_force = force;
            my_list->my_marking = marking;
            my_list->link = NULL;
        }
}
```

## Appendix G. *print_vehicle_info.c* Function Source Code

```
#include <stdext.h>
#include <libkillstop.h>
#include "libent_local.h"
#include <veh_appear.h>
/* #include <liblocale.h> */
#include <libclass.h>
#include <libtime.h>
#include <libcoordinates.h>
#include <sys/time.h>
#include <string.h>

int32 print_vehicle_info(int32 Vehicle_ID)
{
        static FILE     *OutFile;
        static int32    OpenFile = 0;
        char            result[20],
                        appearance_string[20],
                        color_string[10];
        uint32          appearance;
        ForceID         force;
        VehicleMarking  marking;
        static int32    tmpr_blue = 0,
                        tmpr_red = 0,
                        num_blue = 0,
                        num_red = 0,
                        nbk = 0,
                        nrk = 0;
        int32           i = 0,
                        heading_degrees,
                        cell;
        float64         pos[XYZC],
                        pitch,
                        roll,
                        speed = 0,
                        heading,
                        lt,
                        ln,
                        z;
        time_t          timep;
        struct  tm      tmm;
        struct  timeval tv;
        struct  timezone tz;

      if (!bk_lev && !rk_lev)
              return 0;

      if (OpenFile == 0)
        {
                OutFile = fopen("VehicleInfo","a");
```

```
/*
fprintf(OutFile,"Color\tURN\tTime\tPosition\tAlt\tSpeed\tHeading\tAppearance\
n"); */

fprintf(OutFile,"Color\tURN\tTime\tAppearance\tNB\tNBK\tNR\tNRK\n");
                OpenFile = 1;
        }

        ent_get_marking(Vehicle_ID,&marking); // Get vehicle's marking
/*       if this is not an individual vehicle (i.e., platoon or larger),
return */
      if (strlen(marking.text) < 6)
        {
/*           fprintf(OutFile,"%s\t",marking.text); */
           return 0;
        }
      if (strchr(marking.text,32) != NULL)
        {
/*           fprintf(OutFile,"%s contains blank\n",marking.text); */
           return 0;
      {

      gettimeofday(&tv,&tz);
      timep = tv.tv_sec;
      tmm = *gmtime(&timep);
      force = ent_get_force_id (Vehicle_ID);

      if ( force == distinguishedForceID && bk_lev > 0 )
        {
            sprintf(color_string,"Blue");
            if (tmpr_blue != tmm.tm_min)
              {
                if (num_blue)
                  if ((100*nbk)/num_blue + 1 > bpct)
                    {
                       fprintf(OutFile,"\n Stop Blue\n\n");
                       main_clean_up();
                     }
                  num_blue = 0;
                  nbk = 0;
              }
            tmpr_blue = tmm.tm_min;
            num_blue = num_blue + 1;

            appearance = ent_get_appearance (Vehicle_ID);
            if (appearance & (vehDestroyed | vehFlaming))
              {
                 strcpy (appearance_string, "K kill");
                 nbk = nbk + 1;
              }
            else if ((appearance & vehFirepowerDisabled) &&
                          (appearance & vehMobilityDisabled))
              {
                 strcpy (appearance_string, "FM kill");
                 if (bk_lev < 4 )
                   nbk = nbk + 1;
              }
```

```
      else if (appearance & vehFirepowerDisabled)
        {
           strcpy (appearance_string, "Fpr kill");
           if (bk_lev < 3 )
             nbk = nbk + 1;
        }
      else if (appearance & vehMobilityDisabled)
        {
          strcpy (appearance_string, "Mbl kill");
          if (bk_lev == 1 )
            nbk = nbk + 1;
        }
      else
        {
          strcpy (appearance_string, "Healthy");
        }

  }
else if ( force == otherForceID && rk_lev > 0 )
  {
    sprintf(color_string,"Red");
    if (tmpr_red != tmm.tm_min)
      {
        if (num_red)
          if ((100*nrk)/num_red + 1 > rpct)
            {
              fprintf(OutFile,"\n Stop Red\n\n");
              main_clean_up();
            }
          num_red = 0;
          nrk = 0;
      }
     tmpr_red = tmm.tm_min;
     num_red = num_red + 1;

     appearance = ent_get_appearance (Vehicle_ID);
     if (appearance & (vehDestroyed | vehFlaming))
       {
         strcpy (appearance_string, "K kill");
         nrk = nrk + 1;
       }
     else if ((appearance & vehFirepowerDisabled) &&
            (appearance & vehMobilityDisabled))
       {
         strcpy (appearance_string, "FM kill");
         if ( rk_lev < 4 )
           nrk = nrk + 1;
       }
     else if (appearance & vehFirepowerDisabled)
       {
         strcpy (appearance_string, "Fpr kill");
         if ( rk_lev < 3 )
           nrk = nrk + 1;
       }
     else if (appearance & vehMobilityDisabled)
       {
         strcpy (appearance_string, "Mbl kill");
```

```
              if ( rk_lev == 1 )
                nrk = nrk + 1;
            }
           else
            {
              strcpy (appearance_string, "Healthy");
            }

      }
    else if ( force == neutralForceID )
      sprintf(color_string,"Green");
    else
      sprintf(color_string,"Black");

/*        ent_get_position_gcs (Vehicle_ID, pos); // Get vehicle's position//
*/
/*        if (!coord_convert (COORD_GCS, (int32) pos[CELL3D], pos[X], pos[Y],
0.0,               COORD_LATLON, &lt, &ln, &z, TRUE)) */
/*                strcpy (result, coord_format_latlon (lt, ln)); */
/*        else  */
/*                strcpy (result, "?"); */
/*                                                                  */
/*        ent_get_orientation_gcscs (Vehicle_ID, cell, &heading, &pitch,
&roll); // Get vehicle's orientation in radians */
/*        heading_degrees = (int32) RAD_TO_DEG (heading); // Convert
vehicle's orientation from radians to degrees */
/*        if (heading_degrees < 0) */
/*        {  */
/*                heading_degrees += 360;  */
/*        } */
/*    speed = ent_get_speed (Vehicle_ID); // Get vehicle's speed */

    if (ent_is_ic (Vehicle_ID))
      {
        sprintf (appearance_string, "%s (%s)", appearance_string,
           (appearance & lfPositionMask) == lfPositionProne ? "Prone" :
           (appearance & lfPositionMask) == lfPositionCrawling ? "Crawling" :
           (appearance & lfPositionMask) == lfPositionSitting ? "Sitting" :
           (appearance & lfPositionMask) == lfPositionCrouching ? "Crouching"
:
           (appearance & lfPositionMask) == lfPositionKneeling ? "Kneeling" :
           (appearance & lfPositionMask) == lfPositionStanding ? "Standing" :
           (appearance & lfPositionMask) == lfPositionWalking ? "Walking" :
           (appearance & lfPositionMask) == lfPositionRunning ? "Running" :
           "Other");
       }

     fprintf(OutFile,"%s\t",color_string);
     fprintf(OutFile,"%s\t",marking.text);
/*        fprintf(OutFile,"%s\t",ctime(&timep)); */
/*        fprintf(OutFile,"%s\t",result); */
/*        fprintf(OutFile,"%.3f\t",ent_get_altitude_agl(Vehicle_ID));*/
/*        fprintf(OutFile,"%.3f\t",speed); */
/*        fprintf(OutFile,"%d\t",heading_degrees); */
     fprintf(OutFile,"%d\t",tmm.tm_min);
     fprintf(OutFile,"%s\t\t",appearance_string);
     fprintf(OutFile,"%d\t",num_blue);
```

```
        fprintf(OutFile,"%d\t",nbk);
        fprintf(OutFile,"%d\t",num_red);
        fprintf(OutFile,"%d\n",nrk);
        return 0;
}
```

INTENTIONALLY LEFT BLANK

## Appendix H. *libentity.h* Header File Source Code Additions

```
.
.
.
/**
 ** Functions relating to global libentity operation
 **
 **/

extern void check_list (int32 vehicle_id);            /* new code */
extern int32 print_vehicle_info(int32 Vehicle_ID);    /* new code */
.
.
.
```

INTENTIONALLY LEFT BLANK

## Appendix I. *ent_tick.c* Source Code Addition

```
.
.
.
    /* Return if tick is suspended */
    if(ent->suspend_tick)
      return;

    check_list  (vehicle_id);                    /* new code */

    if (!(IS_AGGREGATE(ent)))
    {
.
.
.
```

INTENTIONALLY LEFT BLANK

## Appendix J. Sample *VehicleInfo* File

| Color | URN | Time | Appearance | NB | NBK | NR | NRK |
|---|---|---|---|---|---|---|---|
| Blue | 100A11 | 3 | Healthy | 1 | 0 | 0 | 0 |
| Blue | 100A12 | 3 | Healthy | 2 | 0 | 0 | 0 |
| Blue | 100A13 | 3 | Healthy | 3 | 0 | 0 | 0 |
| Blue | 100A14 | 3 | Healthy | 4 | 0 | 0 | 0 |
| Red | 100A11 | 3 | Healthy | 4 | 0 | 1 | 0 |
| Red | 100A12 | 3 | Healthy | 4 | 0 | 2 | 0 |
| Red | 100A13 | 3 | Healthy | 4 | 0 | 3 | 0 |
| Blue | 100A11 | 4 | Healthy | 1 | 0 | 3 | 0 |
| Red | 100A13 | 4 | Healthy | 1 | 0 | 1 | 0 |
| Red | 100A12 | 4 | Healthy | 1 | 0 | 2 | 0 |
| Blue | 100A13 | 4 | Healthy | 2 | 0 | 2 | 0 |
| Red | 100A11 | 4 | Healthy | 2 | 0 | 3 | 0 |
| Blue | 100A12 | 4 | Healthy | 3 | 0 | 3 | 0 |
| Blue | 100A14 | 4 | Healthy | 4 | 0 | 3 | 0 |
| Blue | 100A11 | 5 | Healthy | 1 | 0 | 3 | 0 |
| Blue | 100A14 | 5 | Healthy | 2 | 0 | 3 | 0 |
| Red | 100A13 | 5 | Healthy | 2 | 0 | 1 | 0 |
| Blue | 100A12 | 5 | Healthy | 3 | 0 | 1 | 0 |
| Red | 100A11 | 5 | Healthy | 3 | 0 | 2 | 0 |
| Red | 100A12 | 5 | Healthy | 3 | 0 | 3 | 0 |
| Blue | 100A13 | 5 | Healthy | 4 | 0 | 3 | 0 |
| Blue | 100A11 | 6 | Healthy | 1 | 0 | 3 | 0 |
| Blue | 100A13 | 6 | Healthy | 2 | 0 | 3 | 0 |
| Red | 100A12 | 6 | Healthy | 2 | 0 | 1 | 0 |
| Blue | 100A14 | 6 | Healthy | 3 | 0 | 1 | 0 |
| Red | 100A11 | 6 | Healthy | 3 | 0 | 2 | 0 |
| Red | 100A13 | 6 | Healthy | 3 | 0 | 3 | 0 |
| Blue | 100A12 | 6 | Healthy | 4 | 0 | 3 | 0 |
| Blue | 100A11 | 7 | Healthy | 1 | 0 | 3 | 0 |
| Red | 100A13 | 7 | Healthy | 1 | 0 | 1 | 0 |
| Blue | 100A12 | 7 | Healthy | 2 | 0 | 1 | 0 |
| Blue | 100A13 | 7 | Healthy | 3 | 0 | 1 | 0 |
| Red | 100A11 | 7 | K kill | 3 | 0 | 2 | 1 |
| Red | 100A12 | 7 | K kill | 3 | 0 | 3 | 2 |
| Blue | 100A14 | 7 | Healthy | 4 | 0 | 3 | 2 |
| Blue | 100A11 | 8 | Healthy | 1 | 0 | 3 | 2 |
| Blue | 100A14 | 8 | Healthy | 2 | 0 | 3 | 2 |

Stop Red

INTENTIONALLY LEFT BLANK

## Appendix K. File *GNUmakefile.in*

```
srcdir=@srcdir@
COMPRESS_DATA=@COMPRESS_DATA@
GZIP=@GZIP@
VPATH=@srcdir@
top_srcdir=@top_srcdir@
CC=@CC@
CXX=@CXX@
CPPFLAGS=@CPPFLAGS@
CFLAGS=@CFLAGS@
CXXFLAGS=@CXXFLAGS@
AR=@AR@
ARFLAGS=@ARFLAGS@
CXX_AR=@CXX_AR@
CXX_ARFLAGS=@CXX_ARFLAGS@
RANLIB=@RANLIB@
RLFLAGS=@RLFLAGS@
LDFLAGS=@LDFLAGS@
LIBS=@LIBS@
JREDIR=@JREDIR@
JREHOME=@JREHOME@
JAVAC=@JAVAC@
JAVAH=@JAVAH@
JAVACFLAGS=@JAVACFLAGS@
JAVAHFLAGS=@JAVAHFLAGS@
JAVAIFLAGS=@JAVAIFLAGS@
JAVACLASSES=@JAVACLASSES@
JAVA_SAF=@JAVA_SAF@

targetroot=@targetroot@
toolsbindir=@toolsbindir@

OBJECTS = \
ent_wrapper.o \
ent_class.o \
ent_coord.o \
ent_depend.o \
ent_init.o \
ent_params.o \
ent_pdus.o \
ent_rva.o \
check_list.o \
print_vehicle_info.o \
ent_tick.o \
ent_update.o \
ent_set.o \
ent_get.o \
ent_event.o

LIBNAME = entity
HEADERS = libentity.h libkillstop.h
```

```
LOCAL_HEADERS = libent_local.h
SAF_MODEL = SM_Entity
PROTO_PREFIX = ENTITY
TYPES = ent.tdl
EXTRA_CLEAN_FILES = ent.tdl

READERS = \
ent_artics.rdr \
ent_timers.rdr

FUNCTION_PREFIX = ent_

JAVA_WRAPPERS = EntityWrapper.java

include $(top_srcdir)/makeinclude/make.librules
```

## Appendix L. Sample Parser File

```
scenario load rda4.1.gz
run 1.0 0
```

INTENTIONALLY LEFT BLANK

## Appendix M. Controller File for OTBSAF Simulations

```
cd /g1/wab/andy/OTB.Feb03/src/OTBSAF

HERE=`pwd`
echo $HERE
for i in 1 2 3
do

  $HERE/otbsaf -nogui -nonet -parser -sourcefile $HERE/parsefile.in -
run_duration 720000 -bpct 50 -rpct 50 -bk_lev 3 -rk_lev 3 >$HERE/outs/out$i
  echo >> $HERE/outs/out$i
  echo >> $HERE/outs/out$i  "  otbsaf run$i ended " >>$HERE/outs/out$i
  echo >> $HERE/outs/out$i
  mv VehicleInfo $HERE/outs/VehicleInfo$i

done
```

INTENTIONALLY LEFT BLANK

## Appendix N. Sample Simulation Output File

```
OTBSAF OTBSAF Version 1.0
Process ID: 29040
OTBSAF built on gawain.arl.army.mil - (Linux 2.4.21-27.0.1.EL) at Tue Jan 11
13:00:32 EST 2005 by andy

                            Network: Off
                    Packet Tee Port: 0
                    Synchronous UDP: True
                                DIS: True
                        DIS Version: 4
                       DIS UDP Port: 3000
                    Bundle DIS PDUs: False
                      Multicast TTL: 32
                    Unicast Address: (null)
  Articulation Dead Reckoning: False
  Simulation Address Override: 0 0
          Send Stealth ESPDUs: False
    Pktvalve Buffer Pool Size: 8192
                 Body Centroid: False
               Terrain Database: knox-0311
                  TDB Directory: ../../terrain
                       GCS Cell: none
                     TDB Memory: 10
             TDB Integrity Check: True
           Blue Kill Percentage: 50
           Red Kill Percentage: 50
               Blue Kill Level: 3
                Red Kill Level: 3
                  Run Duration: 720000
                           GUI: Off
                  GUI warnings: Off
                      Activate: Off
                      Simulate: On
               PO Send Enabled: True
                   Database ID: 1
        Monitor log directory: ../../logs/monitor
                     Benchmark: 0
                Zoom Benchmark: Off
           Default Competence: 0.500000
       Vulnerability Modifier: 1.000000
             Stealth Previews: 0
              Multicast Agent: False
         Absolute Timestamps: False
             Trust Timestamps: False
       Vehicle Loading Factor: 0.010000
              Views Directory: ../../views
                    View file: (null)
          Template Directory: ../../templates
               Template file: (null)
      Standard Load Directory: ../../stdloads
```

```
                          Data Directory: ../../data
                Shared Object Directory: ../../lib
             Dump scheduler information: False
                          Oversize Cursor: On
                             Use RouteMap: True
                           Memory Monitor: Off
                Use ModStealth Protocol: True
                              Use Parser: True
                           Use sourcefile:
/g1/wab/andy/OTB.Feb03/src/OTBSAF/parsefile.in
                 Send StatusChange PDUs: False
                   Send TgtAcq VVA PDUs: False
                   Send DelAcc VVA PDUs: False
                    Send DfDam VVA PDUs: False
                    Send IfDam VVA PDUs: False
                    MKill cants vehicle: True
                       FKill droops gun: True
                     Best Matching Ammo: True
       Detect floating point exception: False
                 Relative Battle Scheme: Off
                      Modify Competence: True
        Force non-constant environment: False
                  Disable dust behavior: True
         Enable phenomenology behavior: True
         Enable environmental mobility: False
       Environmental Weather Simulator: False
     Environmental Sea State Simulator: False
              Gridded Weather Simulator: False
               Environmental Statistics: False
             Smoke Cloud Representation: 2
                  Environment Demo Mode: False
             Dynamic Terrain Operations: False
                          DT Simulator: False
                             DT Scribe: False
               Scribe Backup File Path: ../../logs
                    Scribe Backup File: scribe[exercise_id]
    Use of Existing Scribe Backup File: discard
          New Subscriber initial pause: 1000
            New Subscriber burst pause: 1
             Mines Orientation Present: False
            Mines Burial Depth Present: False
             Mines Temperature Present: False
                       Enable AF model: False
                                 ASPDU: On
                             Migration: On
              Network Monitor Interval: -1
               Network Monitor Address: fff.fff.fff.fff
                           Simple IFF: False
                    Supply Consumption: False
                       Random Failures: False
                           Send Alerts: False
      Thinned TDB Switching Zoom Level: 50000
                  Draw MES Distinctly: False
 Open Agent Architecture (Command Talk): Off
                       Open Agent Host: localhost
                       Open Agent Port: 6666
                Open Agent Server Type: True
```

```
                    Use Ordnance Server: False
       Print Extended Version/Build Info: False
                         Stow Units Only: False
         Block Sending of Emissions PDUs: none
                       KKill detonation: False
              Terrain has contamination: False
                         Test Procedures: (null)
                             Random Seed: 0
                       Enable Async Time: False
                         Repeatable Mode: False
                 RWA Model high fidelity: True
   Enable building bounding box bundling: False
                           Load Scenario: Off
                     Checkpoint Exercise: Off
                      Iconify ModSAF GUI: Off

Reading terrain: knox-0311...

Database Knox-0311 created Wed Dec 18 11:18:08 1996
-------------------------------------------------------
Terrain Format 7 with the following features:
UTM flat, MIXED TIN & GRID POST,
-------------------------------------------------------
Grid Spacing     (METERS): 125
Fixed point basis        : 0.01907349
Origin at 4155000N 545000E in UTM zone 16S (datum WGS84)
Minimum (SW Corner)(X,Y) : (      0.00,       0.00)
Maximum (NE Corner)(X,Y) : (  75000.00,   50000.00)
Minimum Elevation        : (      0.00)
Maximum Elevation        : (    306.99)
-------------------------------------------------------
2729 nodes (86KB), 3440 edges (189KB), 33962 abstract data (133KB)

Successfully read 1 cell.

Loading precomputed routemap file ../../terrain/knox-0311.rnl...
done.
Using a default of site:29040 host:5303.
Pktvalve allocating a pool size of 8192 buffers
Environment: Skipping environment.rdr.
             (No initialized models will be ignored.)
Reading indirect fire delivery accuracy file "cmbt_ifdata.rdr"...
Reading indirect-fire ICM file  "ifdam_icm.rdr"...
Reading indirect-fire HE file   "ifdam_he.rdr"...
Reading indirect-fire damage in "ifdam.rdr"...
Max cutoff for indirect fire detonations is 300 meters.
Using data in physdb for inherent contrast.
Reading protocol conversion rules...

Reading model configuration files from ../../data/entities/modellist.rdr...
.........................................................
.........................................................
.........................................................
.....................................
Successfully read 281 of 281 model configuration files!

Reading dtoconst.rdr from ../../data
```

```
Initializing DTAgent for CTDB
Warning: DTSim failed to read d3b database file
Running in normal time mode.
Warning: Failed to initialize libmso.
Warning: Cultural Features will NOT be sensed.
OTBSAF @ GAWAIN> Sourcing file
/g1/wab/andy/OTB.Feb03/src/OTBSAF/parsefile.in...
1:scenario load rda4.1.gz
Reading scenario file
Loading Scenario
Loading portable scenario:
    Created: "GAWAIN"
    SAF Version: "OTBSAF OTBSAF Version 1.0"
Loading module SM_URadarSectors
Loading module SM_UReactObst
Loading module SM_UReactIF
Loading module SM_UReactSmoke
Loading module SM_UReactAir
Loading module SM_VReceiveRepair
Loading module SM_VReceive
Loading module SM_VCollide
Loading module SM_VOPReactAir
Loading module SM_VMMCM
       Total Objects: 283
           Processed: 283
             Damaged: 0
           Corrected: 0
             Created: 283
OTBSAF @ GAWAIN>
2:run 1.0 0
Running in real time
OTBSAF @ GAWAIN...finished sourcing file
/g1/wab/andy/OTB.Feb03/src/OTBSAF/parsefile.in.

OTBSAF @ GAWAIN> Reading direct fire damage mapping file
"dfdam_mf_M1A2.rdr"...
Will cache Fire PDUs for damage from "munition_US_MX943_submun"
Reading mine damage file "dfdam_M1_mines.rdr"...
Reading direct fire delivery accuracy file "bgun_2A001.rdr"...
Reading direct fire delivery accuracy file "bgun_2A002.rdr"...
Reading direct fire delivery accuracy file "bgun_2A004.rdr"...
Reading direct fire delivery accuracy file "bgun_AGL.rdr"...
Reading direct fire delivery accuracy file "bgun_AGL.rdr"...
Reading direct fire delivery accuracy file "bgun_AGL.rdr"...
Reading direct fire delivery accuracy file "bgun_AGL.rdr"...
Reading direct fire delivery accuracy file "bgun_2A004.rdr"...
Reading direct fire damage mapping file "dfdam_mf_T72M.rdr"...
Reading mine damage file "dfdam_USSR_T72_mines.rdr"...
Reading direct fire delivery accuracy file "bgun_3A002.rdr"...
Reading direct fire delivery accuracy file "bgun_3A003.rdr"...
Reading direct fire delivery accuracy file "bgun_3A006.rdr"...
Reading direct fire delivery accuracy file "bgun_3A005.rdr"...

  otbsaf run1 ended
```

NO. OF COPIES ORGANIZATION

* ADMINISTRATOR
DEFENSE TECHNICAL INFO CTR
ATTN DTIC OCA
8725 JOHN J KINGMAN RD STE 0944
FT BELVOIR VA 22060-6218
*pdf file only

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN IMNE ALC IMS MAIL & REC MGMT
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL CI OK TL TECH LIB
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIR OF COMBAT DEVELOPMENT
ATTN ATZK FD W MEINSHAUSEN
BLDG 1002 ROOM 326
1ST CAVALRY DIV RD
FT KNOX KY 40121-9142

1 CDR US TACOM-ARDEC
ATTN AMSTA AR FSS J WALSH
PICATINNY ARSENAL NJ 07806-5000

1 TECHNOLOGY SVC ASSOC INC
ATTN JON NIDA
3361 ROUSE RD STE 240
ORLANDO FL 32817

2 CDR TRADOC
ATTN ATINZA R REUSS
ATIN I C GREEN
BLDG 133
FT MONROE VA 23651

1 OFC OF THE SECY OF DEFENSE
CTR FOR COUNTERMEASURES
ATTN M A SCHUCK
WHITE SANDS MISSILE RANGE NM 88002-5519

2 CDR US ARMY ARMOR CTR & FT KNOX
ATTN TSM/ABRAMS COL D SZYDLOSKI
DIR UAMBL COL J HUGHES
FORT KNOX KY 40121

2 CDR US TACOM-ARDEC
ATTN AMSTA AR TD J HEDDERICK
B MADECK
PICATINNY ARSENAL NJ 07806-5000

NO. OF COPIES ORGANIZATION

3 CDR US TACOM-ARDEC
ATTN AMSTA AR FSP G A PEZZANO
R SHORR
AMSTA AR FSP I R COLLETT
PICATINNY ARSENAL NJ 07806-5000

3 CDR US TACOM-ARDEC
ATTN AMSTA AR CCH A M PALTHINGAL
E LOGSDON M YOUNG
PICATINNY ARSENAL NJ 07806-5000

1 US MILITARY ACADEMY
MATH SCIENCES CTR OF EXC
ATTN MDN MATH LTC LAMBERT
THAYER HALL
WEST POINT NY 10996-1786

1 CDR US ARMY MMBL
ATTN MAJ J BURNS
BLDG 2021
BLACKHORSE REGIMENT DR
FT KNOX KY 40121

1 CDR ARMY RSCH OFC
4300 S MIAMI BLVD
RSCH TRIANGLE PK NC 27709

1 CDR US ARMY PEO STRI
ATTN J STAHL
12350 RSCH PKWAY
ORLANDO FL 32826-3726

1 CDR US ARMY TRADOC
BATTLE LAB INTEGRATION 7 TECH DIR
ATTN ATCD B J A KLEVECZ
FT MONROE VA 23651-5850

1 OFC OF THE PROJECT MGR
MANEUVER AMMUNITION SYSTEMS
ATTN S BARRIERES
BLDG 354
PICATINNY ARSENAL NJ 07806-5000

1 CDR US ARMY TRADOC ANALYSIS CTR
ATTN ATRC WBA J GALLOWAY
WHITE SANDS MISSILE RANGE NM 88002

1 CDR USAAMC
DEPUTY G3 CURRENT OPERATIONS
ATTN N BIAMON
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

NO. OF
COPIES ORGANIZATION

1 CDR USAARDEC
ATTN AMSRD AAR AE COL P JANKER
BLDG 94
PICATINNY NJ 07806-5000

1 PEO SOLDIER
ATTN C TAMEZ
5901 PUTNAM ROAD
BLDG 328
FT BELVOIR VA 22060-5422

ABERDEEN PROVING GROUND

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL CI OK (TECH LIB)
BLDG 4600

2 CDR US ARMY TECOM
ATTN AMSTE CD B SIMMONS
AMSTE CD M R COZBY
RYAN BLDG

4 DIR US AMSAA
ATTN AMXSY D D SHAEFFER
W BROOKS
AMXSY CA G DRAKE/S FRANKLIN
BLDG 327

1 CDR US ATC
ATTN CSTE AEC COL BROWN
BLDG 400

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM J SMITH
T ROSENBERGER
BLDG 4600

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM B J MORRIS
BLDG 4600

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM BA D LYONS
AMSRD ARL WM BD B FORCH
BLDG 4600

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM BC P PLOSTINS
BLDG 390

NO. OF
COPIES ORGANIZATION

7 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM BF
R ANDERSON P BUTLER
M BARANOSKI W OBERLE
C PATTERSON J WALL
S WILKERSON
BLDG 390

5 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM B VANLANDINGHAM
AMSRD ARL WM MB DOWDING
AMSRD ARL WM MC M MAHER
AMSRD ARL WM MD W ROY
AMSRD ARL WM MA S MCKNIGHT
BLDG 4600

3 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM T P BAKER
AMSRD ARL WM TC R COATES
AMSRD ARL WM TB R SKAGGS
BLDG 309

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM TD SCHOENFELD
BLDG 4600

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM TE B RINGERS
BLDG 1116A

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM T M ZOLTOSKI
BLDG 393